

INICIAR SESIÓN

NUESTROS PLANES

TODOS LOS
CURSOS

FORMACIONES

CURSOS

PARA
EMPRESAS

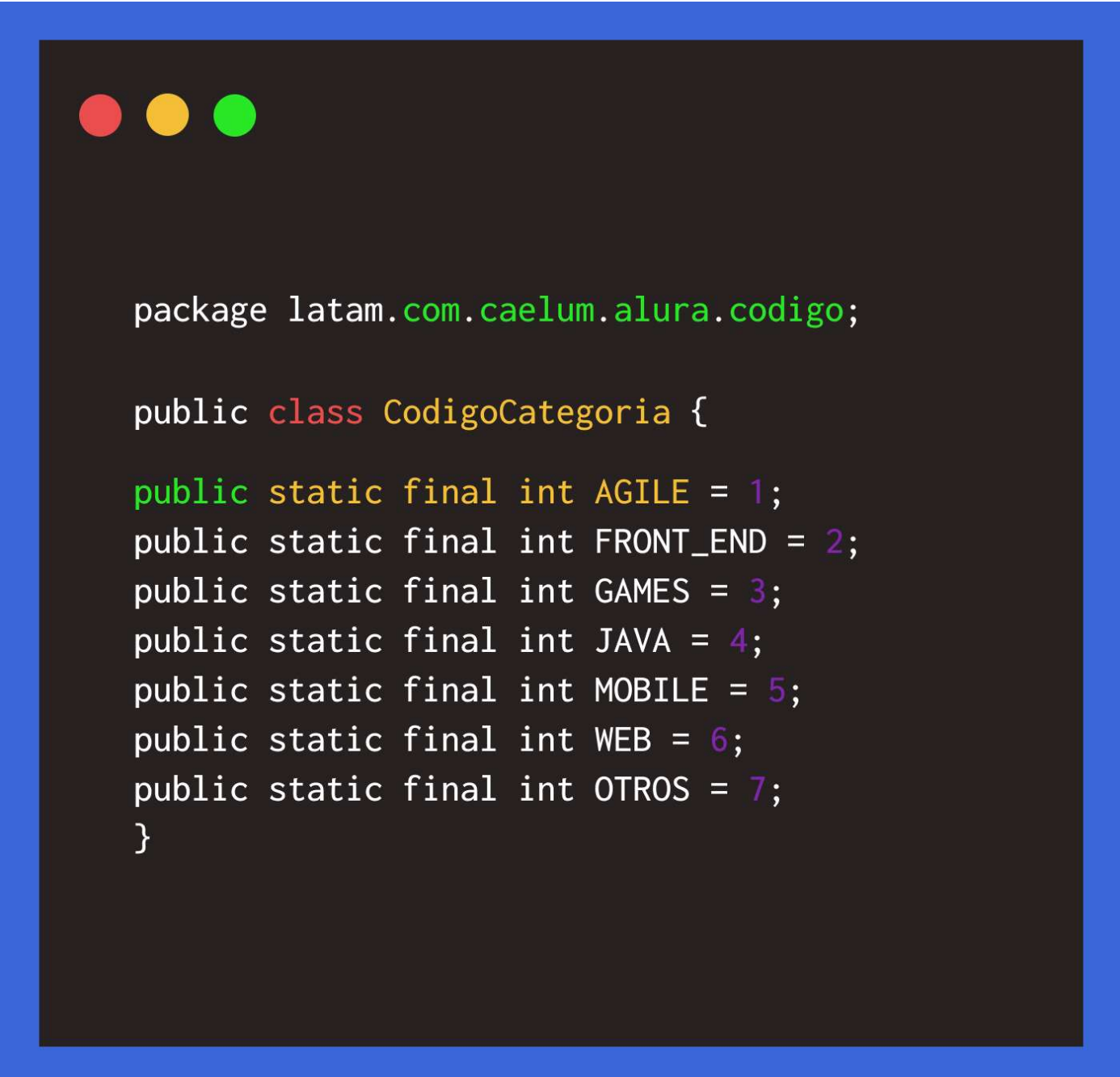
ARTÍCULOS DE TECNOLOGÍA > PROGRAMACIÓN

Como hacer un import static en java



Alex Felipe

09/09/2022



```
package latam.com.caelum.alura.codigo;

public class CategoriaCodigo {

    public static final int AGILE = 1;
    public static final int FRONT_END = 2;
    public static final int GAMES = 3;
    public static final int JAVA = 4;
    public static final int MOBILE = 5;
    public static final int WEB = 6;
    public static final int OTROS = 7;
}
```

Todos los libros de mi sistema necesitan de un nombre, autor y un código de categoría. Para identificar el código de categoría yo tengo la clase CategoriaCodigo en el paquete latam.com.caelum.alura.codigo:

```
package latam.com.caelum.alura.codigo;

public class CategoriaCodigo {
    public static final int AGILE = 1;
    public static final int FRONT_END = 2;
    public static final int GAMES = 3;
    public static final int JAVA = 4;
    public static final int MOBILE = 5;
```

```
public static final int WEB = 6;
public static final int OTROS = 7;
}
```

Y tambien tengo mi clase Libro en el paquete *latam.com.caelum.alura.model*:

```
package latam.com.caelum.alura.model;

public class Libro {
    private String nombre;
    private String autor;
    private int codigoCategoria;

    public Libro(String nombre, String autor, int codigo) {
        this.nombre = nombre;
        this.autor = autor;
        this.codigoCategoria = codigo;
    }

    //metodos
}
```

Cierto, ahora voy a crear un libro nuevo en la clase Main en el paquete *latam.com.caelum.alura.main*:

```
package latam.com.caelum.alura.main;

public class Main { public static void main(String[] args) {
    Libro nuevo = new Libro();
}
}
```

Uy, no compila... Como podemos ver, la clase Libro y la clase Main están en paquetes diferentes, entonces necesitamos importar la clase Libro.

```
package latam.com.caelum.alura.main;
import latam.com.caelum.alura.model.Libro;

public class Main { public static void main(String[] args) {
    Libro nuevo = new Libro("Libro de Java", "Juan Silva", /\* cuál código???
```

Yo no sé cuál es el código para libros de Java, entonces voy a pedir la clase `CodigoCategoria`. Vamos a importarla:

```
package latam.com.caelum.alura.main;
import latm.com.caelum.alura.model.Libro;
import latam.com.caelum.alura.codigo.CodigoCategoria;

public class Main { public static void main(String[] args) {
    Libro nuevo = new Libro("Libro de Java", "Juan Silva", CodigoCategoria.JAV
System.out.println(nuevo); } }
```

Verificando las informaciones del libro:

```
> nombre: Libro de Java autor: Juan Silva codigoCategoria: 4
```

¡Muy bien! logré crear mi libro, pero ahora quiero hacer una lista de libros:

```
//package e imports
import java.util.ArrayList;
import java.util.List;

public class Main { public static void main(String[] args) {
    Libro libro1 = new Libro("Libro de Java", "Juan Silva", CodigoCategoria.JA
    Libro libro3 = new Libro("HTML y CSS", "Rodrigo Teles", CodigoCategoria.FR

    List<Libro> libros = new ArrayList<Libro>();
```

```
libros.addAll(Arrays.asList(libro1, libro2, libro3)); }  
  
}
```

Yo puedo crear la cantidad de libros que desee sin preocuparme con el código de la categoría; sin embargo, hay un detalle... Mire que todas las veces que queremos un código, siempre necesitamos utilizar el prefijo "CodigoCategoria.", no sería simplemente mejor digitar JAVA y ¿la clase ya saber cuál es el código? Ya que todas las constantes de la clase CodigoCategoria SON *static*, podemos hacer el *import static*. ¡Cierto! de la misma forma que en los métodos, siempre usamos el *static* antes, entonces, haremos *static import* también:

```
static import latam.com.caelum.alura.codigo.CodigoCategoria.JAVA;
```

¡Ups! No compila! Para hacer **cualquier import, SIEMPRE** escriba la palabra import antes:

```
//package e imports  
import static latam.com.caelum.alura.codigo.CodigoCategoria.JAVA;  
import static latam.com.caelum.alura.codigo.CodigoCategoria.MOBILE;  
import static latam.com.caelum.alura.codigo.CodigoCategoria.FRONT_END;  
  
public class Main { public static void main(String[] args) {  
    Libro libro1 = new Libro("Libro de Java", "Juan Silva", JAVA);  
    Libro libro2 = new Libro("Libro de Android", MOBILE);  
    Libro libro3 = new Libro("HTML y CSS", "Rodrigo Teles", FRONT_END); //codi  
  
}  
  
}
```

Resolvemos el problema, pero ahora yo tendré que digitar una a una... ¡Qué broma! muy malo esto, ¿cierto? de la misma forma que importamos más de una clase usando el "*", podemos importar todas las constantes estáticas de la clase CodigoCategoria:

```
//package e imports import static latam.com.caelum.alura.codigo.CodigoCategori  
  
public class Main { public static void main(String[] args) {  
    Libro libro1 = new Libro("Libro de Java", "Juan Silva", JAVA);  
    Libro libro2 = new Libro("Libro de Android", MOBILE);  
    Libro libro3 = new Libro("HTML y CSS", "Rodrigo Teles", FRONT_END); //codi  
  
}  
  
}
```



¿

¡El código continúa funcionando de la misma forma! ¿Y si necesitamos de otra categoría? Basta llamar por el nombre de la categoría! Así de simple.

¿Y ahí, te gustó el import estático? ¿Quieres aprender más a fondo el lenguaje Java? Pensando en eso se creó la [formación Java](#) que aborda todas las peculiaridades del lenguaje Java con la intención de preparar el alumno para la certificación.



Alex Felipe Victor Vieira

Es instructor y desarrollador, posee experiencia en Java, Kotlin, Android. Creador de más de 40 cursos, como Kotlin, Flutter, Android, persistencia de datos, comunicación con Web API, personalización de pantallas, tests automatizados, arquitectura de Apps y Firebase. Es experto en Programación Orientada a Objetos, siempre enfocado en compartir las buenas prácticas y tendencias del mercado de desarrollo de software. Trabajó 2 años como editor de contenido en el blog de Alura y hoy en día, aún escribe artículos técnicos.

Traducido por: **Luzdalis Lopez**

Cursos de Programación

ARTÍCULOS DE TECNOLOGÍA > PROGRAMACIÓN

**En Alura encontrarás variados cursos sobre Programación.
¡Comienza ahora!**

SEMESTRAL

US\$49,90

un solo pago de US\$49,90

- ✓ 218 cursos
- ✓ Videos y actividades 100% en Español
- ✓ Certificado de participación
- ✓ Estudia las 24 horas, los 7 días de la semana
- ✓ Foro y comunidad exclusiva para resolver tus dudas
- ✓ Acceso a todo el contenido de la plataforma por 6 meses

¡QUIERO EMPEZAR A ESTUDIAR!

[Paga en moneda local en los siguientes países](#)

ANUAL

US\$79,90

un solo pago de US\$79,90

- ✓ 218 cursos
- ✓ Videos y actividades 100% en Español
- ✓ Certificado de participación
- ✓ Estudia las 24 horas, los 7 días de la semana
- ✓ Foro y comunidad exclusiva para resolver tus dudas
- ✓ Acceso a todo el contenido de la plataforma por 12 meses

¡QUIERO EMPEZAR A ESTUDIAR!

[Paga en moneda local en los siguientes países](#)

Acceso a todos
los cursos

Estudia las 24 horas,
dónde y cuándo quieras

Nuevos cursos
cada semana

NAVEGACIÓN

PLANES

INSTRUCTORES

BLOG

POLÍTICA DE PRIVACIDAD

TÉRMINOS DE USO

SOBRE NOSOTROS

PREGUNTAS FRECUENTES

¡CONTÁCTANOS!

¡QUIERO ENTRAR EN CONTACTO!

BLOG

PROGRAMACIÓN

FRONT END

DATA SCIENCE

INNOVACIÓN Y GESTIÓN

DEVOPS

AOVS Sistemas de Informática S.A
CNPJ 05.555.382/0001-33

SÍGUENOS EN NUESTRAS REDES SOCIALES



ALIADOS



En Alura somos unas de las Scale-Ups seleccionadas por Endeavor, programa de aceleración de las empresas que más crecen en el país.



Fuimos unas de las 7 startups seleccionadas por Google For Startups en participar del programa Growth Academy en 2021

POWERED BY

CURSOS

Cursos de Programación

Lógica de Programación | Java

Cursos de Front End

HTML y CSS | JavaScript | React

Cursos de Data Science

Data Science | Machine Learning | Excel | Base de Datos | Data Visualization | Estadística

Cursos de DevOps

Docker | Linux

Cursos de Innovación y Gestión

Productividad y Calidad de Vida | Transformación Ágil | Marketing Analytics |
Liderazgo y Gestión de Equipos | Startups y Emprendimiento